ELSEVIER

Journal of Computational and Applied Mathematics 110 (1999) 187–203

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

A multigrid preconditioner for stabilised discretisations of advection–diffusion problems ☆

A. Ramage *

Department of Mathematics, University of Strathclyde, Livingstone Tower, 26 Richmond Street, Glasgow G1 1XH, UK

Received 30 November 1998; received in revised form 6 May 1999

Abstract

This paper is concerned with the iterative solution of linear systems arising from stabilised discretisations of advection–diffusion problems on stretched finite element grids. Using nonuniform grids of this type leads, in general, to very badly conditioned matrix problems. We therefore consider using GMRES in conjunction with a multigrid (MG) preconditioning strategy. In particular, we show that in order to achieve the grid-size independent convergence which is characteristic of MG methods, it is essential to use an effective stabilisation strategy at each level of the MG structure. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Advection–diffusion; Streamline upwinding; Stabilisation; Multigrid; GMRES

1. Introduction

The question of how to solve advection–diffusion problems accurately and efficiently is an important topic of current numerical analysis research. It is well known that using standard finite element discretisation methods can lead to oscillatory numerical solutions if the underlying grid is not sufficiently refined where the solution is rapidly changing [5]. Many different stabilisation techniques have been proposed to try to resolve this problem. Whatever method is used, it is important for practical reasons that, as well as providing accurate solutions, the method results in a numerical linear algebra problem which can be efficiently solved. In this paper we show that, for the particular case of the widely used streamline diffusion method, preconditioners based on multigrid methods

☆ This work was supported by NATO Collaborative Research Grant CRG 960782 and the Carnegie Trust for the Universities of Scotland.

* Tel.: +44-141-548-3801; fax: +44-141-552-8657.
E-mail address: a.ramage@strath.ac.uk (A. Ramage)

0377-0427/99/$ - see front matter
PII: S0377-0427(99)00234-4

can be used in conjunction with a Krylov-subspace based iterative methods such as GMRES [18] to achieve this end.

We begin by introducing the advection–diffusion problem

$$\begin{aligned} -v\nabla^2 u + \boldsymbol{w}.\nabla u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \delta\Omega \end{aligned} \tag{1}$$

for a scalar variable u and bounded domain $\Omega \subset \mathbb{R}^2$. The vector $\boldsymbol{w}=(w_x, w_y)$ is a given divergence-free vector (i.e., $\nabla.\boldsymbol{w}=0$) representing a convecting wind and v is a small diffusivity parameter. Throughout this paper we will work with two examples of this type:

Problem I.

domain: $\Omega = [0,1] \times [0,1]$,

wind: $w_x = -\dfrac{1}{\sqrt{2}}, \quad w_y = \dfrac{1}{\sqrt{2}}$,

right-hand side: $f = 0$

boundary conditions: $u = \begin{cases} 1 \text{ when } \frac{1}{2} < x \leqslant 1, \ y = 0 \text{ or } x = 1, \\ 0 \text{ elsewhere on } \delta\Omega. \end{cases}$

Problem II.

domain: $\Omega = [0,1] \times [0,1]$,

wind: $w_x = 2(2y-1)(1-(2x-1)^2)$,
$w_y = -2(2x-1)(1-(2y-1)^2)$,

right-hand side: $f = 0$

boundary conditions: $u = \begin{cases} 1 \text{ when } y = 1, \\ 0 \text{ elsewhere on } \delta\Omega. \end{cases}$

Representative solutions on uniform grids of 64×64 elements with $v = 10^{-2}$ are shown in Fig. 1. The first problem has a constant wind inclined at 45° to the vertical causing boundary layers to appear on the left and top of the domain, while the second has a circulating flow which again produces a layer along the top boundary. These are the types of feature which cause problems for numerical solution methods in general.

Applicable finite element discretisations are well documented in the literature (see, for example, [14,16]). Here we give only a basic outline of the particular technique considered here, namely, the streamline diffusion method (see, for example, [10,11]). The standard Galerkin weak finite element formulation of (1) is given by

$$v(\nabla u, \nabla v) + (\boldsymbol{w}.\nabla u, v) = (f, v) \quad \forall v \in V, \tag{2}$$

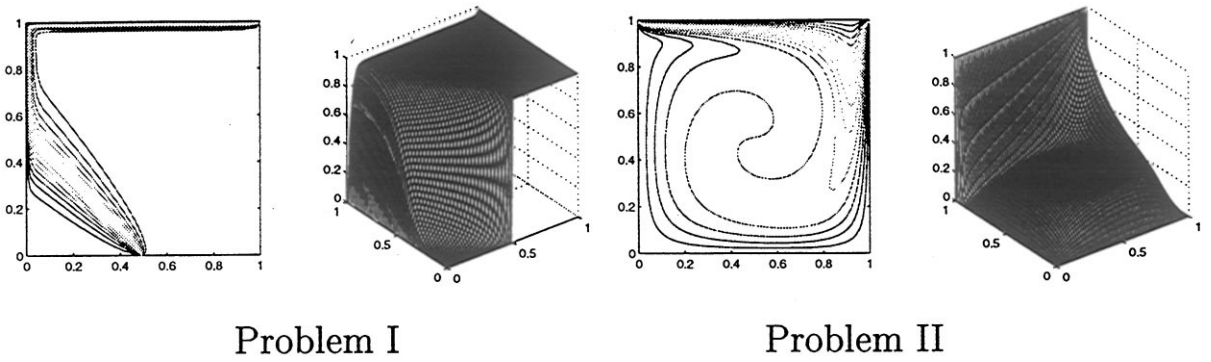

Fig. 1. Sample finite element solutions.

where the test space V is the Sobolev space $\mathscr{H}_0^1(\Omega)$ and $(.,.)$ denotes the standard L_2 inner product. To discretise (2), we introduce a discrete test space $V_h \subset V$ and look for a discrete solution u_h satisfying

$$v(\nabla u_h, \nabla v) + (\boldsymbol{w}.\nabla u_h, v) = (f_h, v) \quad \forall v \in V_h, \tag{3}$$

$$u_h = g_h \quad \text{on } \delta\Omega.$$

Here f_h is the $L^2(\Omega)$ orthogonal projection of f into V_h, g_h is an approximation to g on the boundary, and h is a representative mesh parameter.

Applying a standard Galerkin approximation involves choosing the test functions equal to a set of basis functions $\{\phi_i\}_{i=1}^{\bar{N}}$ for V_h (where the grid has $\bar{N}$ degrees of freedom): here we choose each ϕ_i to be bilinear. Substitution of the discrete solution

$$u_h = \sum_{i=1}^{\bar{N}} U_i \phi_i(x, y), \quad i = 1, \ldots, \bar{N} \tag{4}$$

into (3) then leads to a linear system of equations of the form

$$v \sum_{i=1}^{\bar{N}} U_i(\nabla \phi_i, \nabla \phi_j) + \sum_{i=1}^{\bar{N}} U_i(\boldsymbol{w}.\nabla \phi_i, \phi_j) = (f_h, \phi_j), \quad j = 1, \ldots, \bar{N}.$$

The coefficient matrix of this system can be written as

$$C = v\mathscr{H} + \mathscr{S}, \tag{5}$$

where $\mathscr{H}$ with entries $\mathscr{H}_{ij} = (\nabla \phi_i, \nabla \phi_j)$ is symmetric and positive definite and $\mathscr{S}$ with entries $\mathscr{S}_{ij} = (\boldsymbol{w}.\nabla \phi_i, \phi_j)$ is skew-symmetric.

Unfortunately, for problems which are convection dominated, that is, which have mesh Peclet number

$$\text{Pe} \equiv \frac{h\|\boldsymbol{w}\|}{2v}$$

greater than 1, approximation (3) is unstable and oscillations are observed in solution boundary layers if the underlying grid is not sufficiently fine. One option is to introduce nonuniform meshes which

are constructed specifically to capture such features accurately: these are considered in Section 2. Alternatively, discretisation (3) can be stabilised by introducing some diffusion in the direction of the streamlines. This is done by converting to a Petrov–Galerkin finite element method; that is, we no longer use test and trial functions which are the same. Specifically, the test functions v in (2) are replaced by new test functions

$$v + \beta \boldsymbol{w}.\nabla v, \quad v \in V_h,$$

where β is a stabilisation parameter to be chosen. Although the resulting finite element method is nonconforming, in practice the nonconforming term disappears when it is evaluated in the standard element-wise way as the basis functions are bilinear. The resulting discrete form is therefore

$$\begin{aligned} &\nu(\nabla u_h, \nabla v) + (\boldsymbol{w}.\nabla u_h, v) + (\boldsymbol{w}.\nabla u_h, \beta \boldsymbol{w}.\nabla v) = (f_h, v) + (f_h, \beta \boldsymbol{w}.\nabla v), \\ &u_h = g_h \quad \text{on } \delta\Omega \end{aligned} \tag{6}$$

for all functions $v \in V_h$.

We now comment on the choice of stabilisation parameter β. It can be shown that the solution of (6) satisfies the 'best possible' error estimate (for any degree of polynomial approximation) under the assumption that β in (6) is of the form

$$\beta = \frac{\delta h}{\|\boldsymbol{w}\|} \tag{7}$$

for all $\mathrm{Pe} > 1$, where $\delta > 0$ is a tuning parameter. Note that if the discretised problem is diffusion dominated (i.e., $\mathrm{Pe} \leqslant 1$) then the corresponding 'best' choice above is $\beta = 0$, in which case (6) reduces to the standard Galerkin formulation (3).

Returning to (6), substitution of discrete solution (4) leads to a linear system of equations

$$\begin{aligned} &\nu \sum_{i=1}^{\bar{N}} U_i(\nabla \phi_i, \nabla \phi_j) + \sum_{i=1}^{\bar{N}} U_i(\boldsymbol{w}.\nabla \phi_i, \phi_j) + \frac{\delta h}{\|\boldsymbol{w}\|} \sum_{i=1}^{\bar{N}} U_i(\boldsymbol{w}.\nabla \phi_i, \boldsymbol{w}.\nabla \phi_j) \\ &\quad = (f_h, \phi_j) + \frac{\delta h}{\|\boldsymbol{w}\|}(f_h, \boldsymbol{w}.\nabla \phi_j), \quad j = 1, \ldots, \bar{N} \end{aligned}$$

with coefficient matrix

$$C = \nu \mathscr{H} + \mathscr{S} + \frac{\delta h}{\|\boldsymbol{w}\|} \mathscr{U}. \tag{8}$$

Matrices $\mathscr{H}$ and $\mathscr{S}$ are as in (5), and matrix $\mathscr{U}$ with entries $\mathscr{U}_{ij} = (\boldsymbol{w}.\nabla \phi_i, \boldsymbol{w}.\nabla \phi_j)$ is symmetric and positive definite. We note again that putting $\delta = 0$ gives the Galerkin case.

One obvious and important question is how to choose the tuning parameter δ so that solutions are accurate and cheap to compute. In [4], the authors consider Problem I above with wind $\boldsymbol{w} = (0, 1)$. Based on Fourier analysis of (8) in this case, they conclude that

$$\delta = \frac{1}{2} - \frac{\nu}{h}$$

is a good choice for this model problem in terms of both solution accuracy and efficiency of iterative solution. In addition, they conjecture that for the same problem with a constant wind of unit length in any direction, the generalised parameter

$$\delta_* = \frac{1}{2} - \frac{\nu}{\bar{h}} \tag{9}$$

would be appropriate, where $\bar{h}$ is a measure of the element size in the direction of the wind. Note that the δ used in practice is always positive: if δ_* is less than zero on any particular grid, $\delta = 0$ is used instead as adding streamline diffusion is not necessary.

In this paper we investigate the practical use of the above result for solution of advection–diffusion problems in more general settings. We begin by returning to the idea of modelling boundary layers with nonuniform grids as mentioned above. The particular types of grid used here are described in Section 2. In Section 3, we consider the use of such grids in conjunction with the streamline diffusion method and, in particular, suggest a suitable form for the stabilisation parameter (based on (9)). Unfortunately, although nonuniform grids may enhance solution accuracy, they also introduce difficulties in terms of ill-conditioning of the underlying matrix system. A good preconditioner is therefore required for any standard Krylov subspace iterative solver. In Section 4, we describe the basic multigrid method used to perform this task and use numerical experiments to determine the effect of various stabilisation strategies on the robustness and efficiency of the resulting iterative solver.

2. Nonuniform grids

As stated above, using the simple Galerkin method ($\delta = 0$) on a uniform grid leads to oscillations in the finite element solution if the underlying grid is not sufficiently fine. Two commonly used ways of alleviating this difficulty are either to use a stabilisation technique such as streamline diffusion or to use nonuniform finite element grids which are refined within boundary layers. Here we aim to combine the two approaches by adapting the streamline diffusion technique described in [4] for use on nonuniform grids.

We begin by describing the types of nonuniform grid we will use to model Problems I and II above. As the aim here is to observe general effects of grid stretching, these have not been specifically tailored to the particular problems presented: indeed, the same set of grids is used for both Problems I and II. All grids are tensor products of one-dimensional grids and are therefore topologically square: they have regular connectivity but variable discretisation parameters. The descriptions given below apply to the structure of the one-dimensional component grids with node points given by

$$0 = x_0 < x_1 < \cdots < x_n = 1.$$

The resulting two-dimensional grids have n elements and $\hat{n} = n + 1$ nodes along each side, giving a total of $\bar{N} = \hat{n}^2$ unknowns (before the Dirichlet boundary conditions have been applied). Note that in this paper we do not address the difficulties associated with modelling internal layers like the one present in Problem I (see, for example, [3]).

2.1. Geometrically stretched grids

Choosing variable element lengths of the form

$$h_i = x_i - x_{i-1} = \alpha^{i-1} h_{\min}, \quad i = 1, \ldots, n \tag{10}$$

for some minimum discretisation parameter $h_{\min}$ and stretch factor α leads to a geometrically stretched grid. Such grids are simple to generate and are often used in practice by engineers. The effect of

Table 1
Automatically generated geometric grid stretch factors

No. of elements	8×8	16×16	32×32	64×64
α	1.8393	1.2712	1.1149	1.0534

varying the stretch factor is clear: setting $\alpha = 1$ gives a uniform grid and as α increases, the stretching becomes more pronounced. To assess the performance of the proposed iterative solver, we will solve (1) on a sequence of problems with an increasing number of degrees of freedom, so it is important to choose the stretch factor on each grid in a reasonable way. We will therefore consider two distinct sequences of geometrically stretched grids for each test problem. Note that, in both cases, the actual grids used consist of one stretched grid as described in the interval [0,0.5] (with element lengths increasing) and a reflection of this grid about the line $x = 0.5$ in the interval [0.5,1] (with element lengths decreasing). For the first grid sequence, the stretch factor α is automatically calculated so that the discretisation satisfies

$$h_{\max} = 2h_{\text{reg}}$$

on each top-level grid (of size $n \times n$ elements), where $h_{\text{reg}} = 1/n$ is the grid size of the uniform grid with the same number of elements. The values of α which result are given in Table 1: these grids will be designated by $\alpha = \text{auto}$ in subsequent tables of results. It is clear that as the number of grid points increases, the automatically calculated stretch factor tends to 1 and hence the grids become more uniform as the problem size grows. In case this introduces some 'special' features to the grid sequence, we also consider a grid sequence with (arbitrary) fixed α.

2.2. Adaptively stretched grids

Stretched grids which are adaptively calculated to accurately capture particular features of a specific problem are becoming increasingly common. There are many ways of calculating such grids, based on, for example, a priori knowledge of the exact solution (see, for example, [17]). Another popular idea is the more widely applicable one of calculating a grid adaptively based on the numerical solution; for example, by determining grid points so that they equidistribute a positive function of the numerical solution over the domain (see [8,9]). The resulting grids are often highly nonuniform. Here we consider as an example the family of grids proposed in [1] obtained by equidistribution of a monitor function over the one-dimensional domain. The monitor function in this case is given by

$$M(u(x),x) = \gamma + \left| \frac{\mathrm{d}^2 u}{\mathrm{d}x^2} \right|$$

and is based on the derivative of the solution $u(x)$ of a one-dimensional version of (1) with constant wind. The constant 'floor' value $\gamma > 0$ (which is independent of n) is added to prevent over clustering of nodes in the boundary layer. The resulting grids have two distinct regions: they are exponentially stretched inside the boundary layer and tend to uniform outside the boundary layer. They are similar in idea to Shishkin grids (see, for example, [13]), but the latter grids are uniform in both regions (with a different discretisation parameter in each part). Again, the grids used in practice involve one

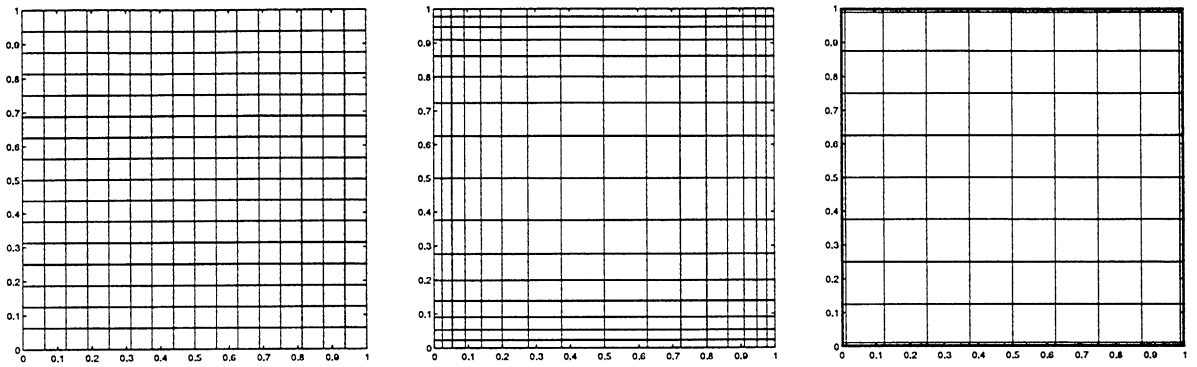

Fig. 2. Sample 16×16 finite element grids with $v = 10^{-3}$.

Table 2
Minimum element size with $v = 10^{-3}$

No. of elements	8×8	16×16	32×32	64×64
Uniform	0.125	0.0625	0.0313	0.0156
Geometric: $\alpha = \text{auto}$	0.0402	0.0233	0.0122	$6.23 \cdot 10^{-3}$
Geometric: $\alpha = 1.2$	0.0931	0.0303	$5.72 \cdot 10^{-3}$	$2.93 \cdot 10^{-4}$
Adaptive	$3.46 \cdot 10^{-4}$	$1.44 \cdot 10^{-4}$	$6.67 \cdot 10^{-5}$	$3.22 \cdot 10^{-5}$

Table 3
Maximum element size with $v = 10^{-3}$

No. of elements	8×8	16×16	32×32	64×64
Uniform	0.125	0.0625	0.0313	0.0156
Geometric: $\alpha = \text{auto}$	0.25	0.125	0.0625	0.0313
Geometric: $\alpha = 1.2$	0.1610	0.1086	0.0881	0.0836
Adaptive	0.25	0.125	0.0625	0.0312

copy of such an adaptive grid in the interval [0,0.5] and its reflection about the line $x = 0.5$ in the interval [0.5,1].

2.3. Grid specifications

Examples of the three grid types of size 16×16 used in the numerical experiments in Section 5 are given in Fig. 2. The diffusivity parameter is $v = 10^{-3}$ (relevant in the adaptive case only). Minimum and maximum element sizes for the full grid sequences used are given in Tables 2 and 3. The minimum element size is of particular importance as it gives an indication of how well a grid will be able to resolve the boundary layer, which is of width $O(v)$. For Problem I above with

Table 4
Mesh Peclet numbers for $v = 10^{-3}$

No. of elements	8×8	16×16	32×32	64×64
Uniform	62.5	31.25	15.625	7.8125

$\|\boldsymbol{w}\| = 1$, the mesh Peclet number for each uniform grid with $v = 10^{-3}$ is also given in Table 4. Note that these are all greater than 1 so we would expect oscillations from the Galerkin method in each case.

3. Choice of stabilisation parameter

For stretched grids, no Fourier analysis such as that in [4] is possible (even for the pure diffusion case). We will, however, use the idea proposed in [4] of choosing δ according to (9) for model problems with constant wind, and analogously set δ here to be

$$\delta_{\mathrm{o}} = \frac{1}{2} - \frac{v}{h_{\mathrm{l}}|\boldsymbol{w}_{\mathrm{l}}|}, \tag{11}$$

where the grid parameter h_{l} and wind $\boldsymbol{w}_{\mathrm{l}} = (W_x, W_y)$ are calculated locally on each element. For nonconstant wind problems, the wind $\boldsymbol{w}_{\mathrm{l}}$ is calculated at the element centre. The local grid parameter h_{l} is then chosen as a measure of the element size in the direction of the local wind, that is,

$$h_{\mathrm{l}} = \begin{cases} h_x & \text{if } W_y = 0, \\ h_y & \text{if } W_x = 0, \\ h_x\sqrt{1 + \left(\frac{W_y}{W_x}\right)^2} & \text{if } |W_x| \geqslant |W_y|, \\ h_y\sqrt{1 + \left(\frac{W_x}{W_y}\right)^2} & \text{if } |W_x| < |W_y|, \end{cases}$$

where h_x and h_y are the element dimensions in the x and y directions, respectively. In practice, if δ_0 is less than zero on an element, $\delta = 0$ is used instead. Examples of the locally calculated δ_0 on a uniform and a stretched grid of size 16×16 for Problem II (with circulating wind) for $v = 10^{-3}$ are shown in Fig. 3. The resulting stabilisation strategy is clearly very different in the two cases.

We now attempt to get an indication of how solution accuracy is affected by changing δ using Problem I with $v = 10^{-3}$. In general, care must be taken when trying to assess the accuracy of the solutions found on nonuniform grids, as many standard node-based measures can be misleading with regard to the size of the error in the boundary layer. For example, a uniform grid will not have any points in the layer if h is greater than v. In addition, as stated above we have made no attempt to find the 'best' grid of each type in any sense for this particular problem: the measure described below should not therefore be seen as fairly reflecting the accuracy of the different grid types.

We compare solutions calculated on all grids to one chosen 'accurate' fine grid solution, namely the solution on a 256×256 version of the adaptive grid. Specifically, we construct the vector $\boldsymbol{u}_{\mathrm{l}}$ (the bilinear interpolant of this fine grid solution at the coarser grid finite element nodes) and examine

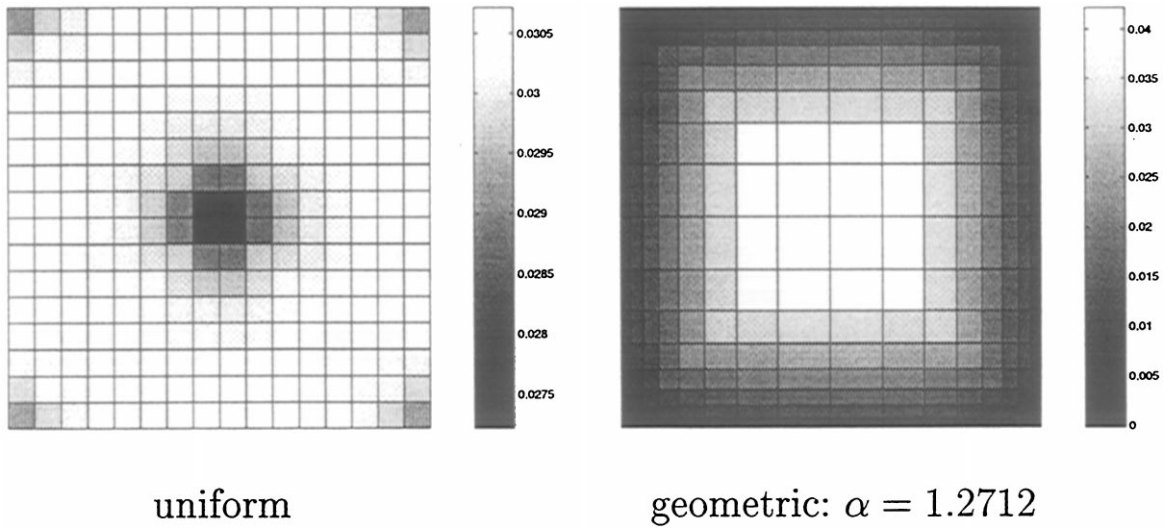

Fig. 3. Sample plot of stabilisation parameter on 16×16 grid. On the uniform grid, $0.027\leqslant\delta\leqslant0.031$; on the stretched grid, $0.006\leqslant\delta\leqslant0.060$.

the discrete L_2 norm of the error vector $\boldsymbol{e} = \boldsymbol{u}_\mathrm{h} - \boldsymbol{u}_\mathrm{I}$ (where $\boldsymbol{u}_\mathrm{h}$ is the finite element nodal solution vector). This is given by

$$\left\{\sum_{i,j=1}^{n-1}\frac{1}{4}(h_i+h_{i+1})(k_j+k_{j+1})|e_{ij}|^2\right\}^{1/2},$$

where h_i and k_j are the grid point spacings in the x and y directions, respectively and e_{ij} is the entry of vector $\boldsymbol{e}$ corresponding to grid point (x_i, y_j). This measure, which is a commonly used criterion for computing solution accuracy (see, for example, [7]), is tabulated for various values of δ in Table 5. We again stress that we are interested only in comparing the effect of changing δ on any particular grid: the errors for $\delta = \delta_\mathrm{o}$ are clearly the smallest of those tabulated in each case.

4. Iterative convergence with a multigrid preconditioner

As stated in the introduction, no matter which discretisation or stabilisation method is employed, it is important to be able to solve the underlying linear system efficiently. A Krylov subspace-type iterative method such as GMRES [18] is therefore appropriate.

Unfortunately, discretising on stretched grids adversely affects the condition of the resulting coefficient matrices, with a corresponding degradation in the performance of GMRES. As an example, estimates of the 1-norm condition numbers (obtained using the MATLAB function `condest` [12]) and unpreconditioned GMRES iteration counts for Problem I with $\nu = 10^{-3}$ are given in Table 6. The situation only gets worse as α increases or as ν decreases, that is, as the geometric and adaptive grid stretching respectively become more pronounced. It is clear that some form of preconditioning is essential.

Table 5
L_2 errors

Grid	8×8	16×16	32×32	64×64
Uniform				
$\delta = 0$	3.29	1.143	$7.921 \cdot 10^{-1}$	$7.211 \cdot 10^{-2}$
$\delta = \delta_o$	$5.957 \cdot 10^{-2}$	$4.406 \cdot 10^{-2}$	$2.316 \cdot 10^{-2}$	$9.815 \cdot 10^{-3}$
$\delta = 0.5$	$7.466 \cdot 10^{-1}$	$7.192 \cdot 10^{-1}$	$7.058 \cdot 10^{-1}$	$6.996 \cdot 10^{-1}$
Geometric: $\alpha = auto$				
$\delta = 0$	2.587	1.065	$8.272 \cdot 10^{-1}$	$7.844 \cdot 10^{-1}$
$\delta = \delta_o$	$2.462 \cdot 10^{-1}$	$1.560 \cdot 10^{-1}$	$9.278 \cdot 10^{-2}$	$6.424 \cdot 10^{-2}$
$\delta = 0.5$	$8.858 \cdot 10^{-1}$	$8.305 \cdot 10^{-1}$	$8.029 \cdot 10^{-1}$	$7.782 \cdot 10^{-1}$
Adaptive				
$\delta = 0$	1.07	1.038	1.017	1.005
$\delta = \delta_o$	$1.504 \cdot 10^{-1}$	$1.189 \cdot 10^{-1}$	$8.699 \cdot 10^{-2}$	$5.676 \cdot 10^{-2}$
$\delta = 0.5$	1.075	1.040	1.017	1.005

Table 6
Coefficient matrix condition numbers and GMRES iteration counts

No. of elements	8×8	16×16	32×32	64×64
Condition numbers				
Uniform	87.45	397.4	$1.6 \cdot 10^3$	$6.0 \cdot 10^3$
Geometric: $\alpha =$ auto	83.97	386.6	$1.6 \cdot 10^3$	$6.0 \cdot 10^3$
Geometric: $\alpha = 1.2$	81.47	387.5	$2.6 \cdot 10^3$	$3.0 \cdot 10^5$
Adaptive	$1.3 \cdot 10^4$	$8.7 \cdot 10^4$	$2.8 \cdot 10^5$	$7.1 \cdot 10^5$
GMRES iteration counts				
Uniform	22	31	50	90
Geometric: α=auto	25	47	88	167
Geometric: $\alpha = 1.2$	22	42	136	612
Adaptive	38	135	387	> 1000

Multigrid (MG) methods have been used very successfully as solvers in their own right to achieve grid-size independent convergence in many different areas of numerical analysis, including advection–diffusion problems. We refer the reader to the extensive MG literature for details (see, for example, [2,6,20]) and confine ourselves to a very basic outline here. The idea of using MG as a preconditioner is relatively recent but is increasing in popularity (see, for example, [15,21]). We aim to develop a simple but robust MG preconditioner for advection–diffusion problems on stretched grids.

The essential idea of a multigrid algorithm is straightforward. Basic iterative methods, or smoothers, such as damped Jacobi or Gauss–Seidel (see, for example, [19]) are efficient at reducing rough or short wavelength components of the error on a fine grid. The main MG principle is to approximate the smooth or long wavelength part of the fine grid error on a coarser grid, so that it can in turn be reduced via an application of the smoothing method. Here, for any particular pair of grids, we will denote the fine grid by G_f and the related coarse grid by G_c (with associated vectors $\boldsymbol{u}_f$ and $\boldsymbol{u}_c$). We assume that each grid has an even number of elements along each side so that G_c can be

formed by removing every second node from G_f. To specify a complete MG algorithm, we need several components:

- a cycle structure prescribing in which order the grids will be visited;
- a smoothing technique to reduce high-frequency components on G_f;
- a prolongation matrix P such that $\boldsymbol{u}_f = P\boldsymbol{u}_c$;
- a restriction matrix R such that $\boldsymbol{u}_c = R\boldsymbol{u}_f$;
- a method of approximating the fine grid coefficient matrix A_f on the coarse grid as A_c.

4.1. The MG preconditioner

There are various possible choices for each of the above components.

Cycle structure: As our aim is to keep things as simple as possible, we use the most basic MG structure of a single V-cycle on a series of nested grids. That is, we begin with a top (finest) grid of a certain size, then produce a series of coarser grids down to the coarsest (2×2 elements) level. Information is then passed straight back up through the same sequence in reverse to the finest level. Note that any stretching present will be more pronounced on the coarser grids: for example, a coarse grid constructed from a geometrically stretched grid with stretch factor α will have a stretch factor of α^2, etc.

Smoother: The smoother used is line Gauss–Seidel, with one horizontal (left $\rightarrow$ right) and one vertical (bottom $\rightarrow$ top) sweep per smoothing step (see, for example, [20]). One pre- and one post-smoothing step is applied on each level.

Grid transfer operators: One of the most important parts of any multigrid algorithm is the mechanism by which information is passed between the coarse and fine grids. Here we compare the performance of three types of prolongation.

- *Bilinear interpolation*: Standard bilinear interpolation is easily applied in both the uniform and nonuniform grid cases.
- *Operator-dependent interpolation*: Many prolongation operators have been proposed which use information from the fine grid coefficient matrix A_f. The idea of the first operator-dependent interpolation we consider here (see [20], Section 5.4) arose in the context of diffusion problems with variable coefficients. Such problems are closely linked to diffusion problems on nonuniform grids.

  The idea of the operator is straightforward to illustrate in one dimension. Function values at nodes common to both fine and coarse grids are passed directly. For a fine grid mid-point node i, the usual bilinear interpolation weights on a uniform grid $(1/2, 1/2)$ are replaced by

$$-\frac{A_f(i,i-1)}{A_f(i,i)}, \quad -\frac{A_f(i,i+1)}{A_f(i,i)},$$

  where A_f is the fine grid coefficient matrix.

  In two dimensions, the idea is essentially the same. At points which are common to both grids, function values are transferred directly. For horizontal mid-points, the fine grid value is a weighted sum of the function value on either side, using weights which involve 'lumping' the entries of A_f to give a one-dimensional operator at the mid-points. Vertical mid-points have weights based on

an analogous horizontal averaging process. Finally, the values at the remaining central points are chosen to satisfy

$$A_f P \boldsymbol{u}_c = 0. \tag{12}$$

That is, all previously calculated surrounding function values are used, with the weights again dependent on fine grid matrix A_f. Note that for points where the above calculations are restricted by boundaries, the relevant weights are replaced by the standard bilinear interpolation values of $1/2$.

- *De Zeeuw interpolation*: A new prolongation operator was proposed by de Zeeuw [22] expressly for advection–diffusion problems. Here the weights depend not only on the fine grid matrix stencil but also on the stencils of its symmetric and skew-symmetric parts.

  The resulting prolongation matrix P is formed as follows. Function values at common coarse and fine points are transferred directly as before. For horizontal and vertical mid-points, the weights are based on the symmetric and skew-symmetric parts of A_f (see [22] for details). The values at the fine grid centre points are found with weights as in the operator-dependent case (that is, via Eq. (12)). As above, bilinear interpolation is used where necessary at the boundaries.

Restriction: The restriction matrix, R, is the transpose of the prolongation matrix, P, in each case.

Coarse grid approximation: We compare two standard methods of generating coarse grid approximations at each multigrid level (see, for example, [20], Chapter 6).

- *Discretisation coarse grid approximation* (*DCA*). As the name implies, this involves calculating each coarse grid coefficient matrix by direct discretisation of the PDE on that coarse grid.
- *Galerkin coarse grid approximation* (*GCA*). Here the coarse grid matrix A_c is formed algebraically via the equation

$$A_c = R A_f P. \tag{13}$$

4.2. Iterative convergence

We now present the results of several numerical experiments which implement the algorithms outlined in the previous section for Problems I and II. The initial guess $\boldsymbol{x}_0$ was zero in each case, and the GMRES algorithm (with preconditioning applied on the right) was considered converged when the residual $\boldsymbol{r}_k$ satisfied

$$\frac{\|\boldsymbol{r}_k\|_2}{\|\boldsymbol{r}_0\|_2} \leqslant 10^{-6},$$

where $\boldsymbol{r}_0$ is the initial residual.

Results obtained by applying a MG preconditioned GMRES algorithm of the above type to Problems I and II are presented in Tables 7–12. In each case, iteration counts are given for MG preconditioned GMRES on grids from size 8×8 to 64×64 elements. The four grid sequences used are as stated in Section 2, with results given for the three types of prolongation and two coarse grid operators described above.

Table 7
GMRES iteration counts for Problem I with DCA and $v = 10^{-3}$

	$\delta = \delta_0$				$\delta = \delta_*$			
No. of elements	8×8	16×16	32×32	64×64	8×8	16×16	32×32	64×64
Bilinear interpolation								
Uniform	4	4	4	4	4	4	4	4
Geometric: $\alpha =$ auto	4	4	4	4	9	14	33	71
Geometric: $\alpha = 1.2$	4	4	4	4	4	8	66	> 100
Adaptive	3	4	4	4	11	36	> 100	> 100
Operator-dependent interpolation								
Uniform	4	4	4	4	4	4	4	4
Geometric: $\alpha =$ auto	4	4	5	4	11	24	60	> 100
Geometric: $\alpha = 1.2$	4	4	5	35	5	22	> 100	> 100
Adaptive	3	13	39	86	13	> 100	> 100	> 100
De Zeeuw interpolation								
Uniform	4	4	4	4	4	4	4	4
Geometric: α=auto	3	4	4	4	9	19	38	86
Geometric: $\alpha = 1.2$	4	4	4	4	4	13	100	> 100
Adaptive	3	4	4	4	11	35	> 100	> 100

Table 8
GMRES iteration counts for Problem I with DCA, bilinear interpolation and $v = 10^{-3}$

No. of elements	8×8	16×16	32×32	64×64
$\delta = 0$				
Uniform	50	> 100	> 100	> 100
Geometric: $\alpha =$ auto	50	> 100	> 100	> 100
Geometric: $\alpha = 1.2$	50	> 100	> 100	> 100
Adaptive	11	36	> 100	> 100
$\delta = 0.5$				
Uniform	4	4	4	4
Geometric: α=auto	8	14	31	59
Geometric: $\alpha = 1.2$	4	8	63	> 100
Adaptive	11	36	> 100	> 100

4.2.1. Problem I

Table 7 contains results for Problem I with $v = 10^{-3}$ using discretisation coarse grid approximation. In the left-hand columns, δ has been chosen according to (11) to give an accurate approximation on each finite element grid level. The iteration counts with bilinear and de Zeeuw interpolation show the grid-size independent convergence expected from a MG method, even on the stretched grids. In the right-hand columns, exactly the same MG algorithms are applied on each grid with $\delta = \delta_*$ at each level, where the value of δ_* is given by (9) for the equivalent uniform grid at the finest level. It is immediately clear that the convergence behaviour of all of the methods has deteriorated very badly, with the exception of the uniform grid case. These results are typical of those obtained using

Table 9
GMRES iteration counts for Problem I with GCA and $v = 10^{-3}$

	$\delta = \delta_0$				$\delta = \delta_*$			
No. of elements	8×8	16×16	32×32	64×64	8×8	16×16	32×32	64×64
Bilinear interpolation								
Uniform	4	4	6	20	4	4	6	20
Geometric: $\alpha = \text{auto}$	4	4	5	10	10	18	53	> 100
Geometric: $\alpha = 1.2$	4	4	4	4	4	12	> 100	> 100
Adaptive	3	4	4	4	11	36	> 100	> 100
Operator-dependent interpolation								
Uniform	4	4	4	4	4	4	4	4
Geometric: $\alpha = \text{auto}$	4	4	5	5	9	15	30	57
Geometric: $\alpha = 1.2$	3	4	5	5	4	10	64	> 100
Adaptive	3	3	4	5	11	35	> 100	> 100
De Zeeuw interpolation								
Uniform	4	4	4	4	4	4	4	4
Geometric: $\alpha = \text{auto}$	3	4	4	4	9	16	31	56
Geometric: $\alpha = 1.2$	3	4	4	4	4	10	66	> 100
Adaptive	3	4	4	4	11	35	> 100	> 100

Table 10
GMRES iteration counts for Problem II with DCA and $v = 10^{-2}$

	$\delta = \delta_0$				$\delta = \delta_*$			
No. of elements	8×8	16×16	32×32	64×64	8×8	16×16	32×32	64×64
Bilinear interpolation								
Uniform	4	4	4	3	4	4	5	> 100
Geometric: $\alpha = \text{auto}$	4	4	4	4	7	10	13	> 100
Adaptive	3	4	4	3	7	13	> 100	> 100
Operator-dependent interpolation								
Uniform	5	4	4	4	4	5	6	> 100
Geometric: $\alpha = \text{auto}$	4	4	4	4	8	11	17	> 100
Adaptive	4	5	4	4	8	33	> 100	> 100
De Zeeuw interpolation								
Uniform	4	4	4	3	4	4	4	> 100
Geometric: $\alpha = \text{auto}$	4	4	4	4	7	9	12	> 100
Adaptive	4	4	4	4	7	17	> 100	> 100

different values of fixed δ, including $\delta = 0$ and 0.5. Sample results for bilinear interpolation using these latter values are shown in Table 8.

Exactly the same pattern of results is seen when the discretisation coarse grid approximation is replaced by a Galerkin one. Results for the same problem with MG preconditioner using GCA are given in Table 9, again for $\delta = \delta_o$ and δ_*. They are very similar to those in Table 7, although we note in passing that combining operator-dependent interpolation with DCA or bilinear interpolation

Table 11
GMRES iteration counts for Problem II with GCA and $v = 10^{-2}$

	$\delta = \delta_0$			
No. of elements	8×8	16×16	32×32	64×64
Bilinear interpolation				
Uniform	4	5	8	> 100
Geometric: $\alpha =$ auto	4	4	6	16
Adaptive	4	4	5	7
Operator-dependent interpolation				
Uniform	4	5	5	4
Geometric: $\alpha =$ auto	3	4	4	4
Adaptive	3	4	5	4
De Zeeuw interpolation				
Uniform	4	4	4	3
Geometric: $\alpha =$ auto	3	4	3	4
Adaptive	3	4	4	3

Table 12
GMRES iteration counts for Problem II with $\delta = \delta_0$ and $v = 10^{-3}$

	DCA				GCA			
No. of elements	8×8	16×16	32×32	64×64	8×8	16×16	32×32	64×64
Bilinear interpolation								
Uniform	6	7	7	6	6	8	> 100	> 100
Geometric: $\alpha =$ auto	5	7	7	6	5	8	11	> 100
Adaptive	4	6	7	7	4	6	7	> 100
Operator-dependent interpolation								
Uniform	7	12	11	9	6	9	11	10
Geometric: α=auto	5	7	9	6	5	8	10	10
Adaptive	4	7	12	11	3	6	9	10
De Zeeuw interpolation								
Uniform	6	7	7	6	6	7	7	8
Geometric: α=auto	5	7	6	5	4	6	6	6
Adaptive	4	6	6	5	3	6	7	7

with GCA seems to cause a degradation in performance. More importantly, however, results with GCA again degrade very badly on all stretched grids for fixed δ exactly as in the DCA case. The results for $\delta = 0$, 0.5, etc., again follow a similar pattern to those for $\delta = \delta_*$.

As mentioned above, MG could also be used directly as a solver for these problems rather than as a preconditioner. The results obtained on repeating the above experiments with a pure MG algorithm can be summarised as follows. When MG works well as a preconditioner, for example, for the cases with 3, 4 or 5 GMRES iterations in Table 7, pure MG also works well, converging in essentially the same number of pure MG iterations. However, whenever the GMRES iteration count in Table 7

is more than 5, the equivalent pure MG iteration diverges. That is, use of MG as a solver or preconditioner exhibits the same trends with respect to the choice of stabilisation parameter, interpolation and coarse grid approximation, but overall the GMRES preconditioned algorithm is more robust. We therefore focus on the latter approach.

It is clear that the choice of $\delta = \delta_o$, which gives an accurate discretisation at every MG level, is of critical importance in obtaining grid-independent convergence on nonuniform grids.

4.2.2. Problem II

The same experiments were repeated on the more difficult Problem II with nonconstant circulating wind. Results comparing the various MG preconditioners for Problem II are this time given for two different values of v: for $v = 10^{-2}$ in Tables 10 and 11 and for $v = 10^{-3}$ in Table 12. Because the choice of geometric stretch factor did not have a dramatic effect on the results in the previous example, iteration counts are shown for the grid sequence with automatically calculated stretch factor only.

In terms of selecting the correct upwind parameter, the behaviour was qualitatively the same as for Problem I in both cases; that is, choosing the same constant value of δ for all grid levels gives very poor convergence on the nonuniform grids no matter what actual constant value is chosen, whereas the choice $\delta = \delta_o$ once again gives MG-like convergence. This is illustrated by the sample iteration counts for $\delta = \delta_o$ and δ_* given in Table 10 (for $v = 10^{-2}$). The effectiveness of δ_o as a stabilisation parameter is again independent of the coarse grid approximation used: analogous results for $\delta = \delta_o$ with GCA are shown in Table 11. Note that here the performance with $\delta = \delta_*$ degenerates even on the uniform grid. This may be due to the fact that, for this more difficult problem, the wind direction is not constant.

Although reducing v from 10^{-2} to 10^{-3} leads to an overall increase in iteration counts, convergence is still essentially grid independent when an appropriate stabilisation parameter is used, as shown in Table 12. Results for fixed δ are not shown here as they are again similar in pattern to the preceding examples. The same trends with respect to prolongation performance as in the constant wind case are also visible.

5. Conclusion

The aim of this study has been to construct a multigrid-preconditioned GMRES algorithm which is effective for solving advection–diffusion problems using stabilised discretisations on adaptively stretched finite element grids. The key conclusion is that to obtain the sort of grid-size independent convergence usually associated with multigrid methods, it is crucial to choose a stabilisation parameter δ which is appropriate to the stretched grid at each level of the multigrid structure.

Acknowledgements

The author would like to thank Howard Elman and David Silvester for their participation in helpful discussions.

References

[1] G. Beckett, J.A. Mackenzie, Convergence analysis of finite difference approximations of an inhomogeneous singularly perturbed boundary value problem using grid equidistribution, Mathematics Technical Report 14-97, University of Strathclyde, 1997.

[2] K. Brand, M. Lemke, J. Linden, Multigrid Bibliography, in: S.F. McCormick (Ed.), Multigrid Methods, SIAM, Philadelphia, 1987, pp. 189–230.

[3] H.C. Elman, Y.-T. Shih, Modified streamline diffusion schemes for convection–diffusion problems, Technical Report CS-TR-3835, University of Maryland, 1998, Comput. Methods Appl. Mech. Eng. 174 (1999) 137–151.

[4] B. Fischer, A. Ramage, D.J. Silvester, A.J. Wathen, On parameter choice and iterative convergence for stabilised discretisations of advection–diffusion problems, Mathematics Technical Report 37-96, University of Strathclyde, 1996, Comput. Methods Appl. Mech. Eng. 179 (1999) 185–202.

[5] P.M. Gresho, R.L. Lee, Don't suppress the wiggles — they're telling you something, in: T.J.R. Hughes (Ed.), Finite Element Methods for Convection Dominated Flows, AMD, Vol. 34, ASME, New York, 1979, pp. 37–61.

[6] W. Hackbusch, Multi-Grid Methods and Applications, Springer, New York, 1980.

[7] A.F. Hegarty, E. O'Riordan, M. Stynes, A comparison of uniformly convergent finite difference schemes for two-dimensional convection-diffusion problems, J. Comput. Phys. 105 (1993) 24–32.

[8] W. Huang, Y. Ren, R.D. Russell, Moving mesh partial differential equations based on the equidistribution principle, SIAM J. Numer. Anal. 31 (1994) 709–730.

[9] W. Huang, D.M. Sloan, A simple adaptive grid methods in two dimensions, SIAM J. Sci. Comput. 15 (1994) 776–797.

[10] T.J.R. Hughes, A. Brooks, A multidimensional upwind scheme with no crosswind diffusion, in: T.J.R. Hughes, (Ed.), Finite Element Methods for Convection Dominated Flows, AMD, Vol. 34, ASME, New York, 1979, pp. 120–131.

[11] C. Johnson, The streamline diffusion finite element method for compressible and incompressible fluid flow, in: D.F. Griffiths, G.A. Watson (Eds.), Numerical Analysis 1989, Pitman Research Notes in Mathematics, Longman, London, 1989, pp. 155–181.

[12] MATLAB High Performance Numeric Computation and Visualization Software, The MathWorks, Inc., 24 Prime Park Way, Natik, MA 01760, 1998.

[13] J.J.H. Miller, E. O'Riordan, G.I. Shishkin, On piecewise-uniform meshes for upwind and central difference operators for solving singularly perturbed problems, IMA J. Numer. Anal. 15 (1995) 89–99.

[14] K.W. Morton, Numerical Solution of Convection–Diffusion Problems, Chapman & Hall, London, 1996.

[15] C.W. Oosterlee, T. Washio, An evaluation of parallel Multigrid as a solver and a preconditioner for singularly perturbed problems, SIAM J. Sci. Comput. 19 (1998) 87–110.

[16] A. Quarteroni, A. Valli, Numerical Approximation of Partial Differential Equations, Springer, New York, 1994.

[17] H.-G. Roos, M. Stynes, L. Tobiska, Numerical Methods for Singularly Perturbed Differential Equations, Springer, Berlin, 1996.

[18] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869.

[19] R.S. Varga, Matrix Iterative Analysis, Prentice-Hall, New Jersey, 1962.

[20] P. Wesseling, An Introduction to Multigrid Methods, Wiley, Chichester, 1991.

[21] C.S. Woodward, A Newton-Krylov-Multigrid solver for variably saturated flow problems, Tech. Report UCRL-JC-129371, Lawrence Livermore National Laboratory, Livermore, CA, 1998.

[22] P.M. De Zeeuw, Matrix-dependent prolongations and restrictions in a blackbox multigrid solver, J. Comput. Appl. Math. 33 (1990) 1–27.